

# Efficient Vision Data Processing on a Mobile Device for Indoor Localization

Michał Nowicki, Jan Wietrzykowski, Piotr Skrzypczyński, *Member, IEEE*

**Abstract**—The paper presents an approach to indoor personal localization using a sequence of captured images. The solution is tailored specifically for processing vision data on mobile devices with limited computing power. The presented FastABLE system is based on the ABLE-M place recognition algorithm, but introduces major modifications to the concept of image matching, in order to radically cut down the processing time, and to improve scalability. FastABLE is compared against the original OpenABLE and the popular OpenFABMAP place recognition systems.

## I. INTRODUCTION

### A. Motivation

Personal indoor localization is still a problem to be solved without using additional infrastructure. Although much work has been done on practical indoor localization, resulting in commercially available solutions<sup>1</sup>, there is a need for approaches that are independent from the artificial sources of signals, such as RFID/BLE beacons or WiFi networks. A vast range of applications may benefit from infrastructure-less indoor localization, such as context-aware advertising or personal guidance in public buildings. In this application context a viable solution is to utilize the ubiquitous mobile devices – smartphones or tablets, which are equipped with various sensors and considerable computing power. The ever increasing performance of mobile devices and the availability of high-quality cameras allows us to consider vision data as a source of localization signals. Localization utilizing visual information can operate regardless of particular features encountered in the environment, in a way similar to human perception of the surrounding world.

### B. State of the Art

Application of cameras for localization was already researched extensively in robotics, resulting in a number of Visual Odometry (VO) [1] and Simultaneous Localization and Mapping (SLAM) [2] algorithms. Unfortunately, the use of VO/SLAM on mobile devices is challenging due to the rapid, unconstrained motion of the embedded camera, limited availability of features, and high computing power demanded by VO/SLAM systems. Our own experience [3] proved that it is feasible to implement a VO algorithm on a modern Android mobile device. Among the state-of-the-art visual localization algorithms, PTAM was implemented with limited functionality

on iPhone [4], while SVO [5] was demonstrated on the limited-power Odroid platform. Algorithms designed to operate on less capable processors [6], [7] can be implemented on mobile devices [8], but they still drain a lot of energy. Vision-based tracking was already integrated with WiFi and inertial data for personal localization [9], but using a setup that only resembled a mobile device, thus avoiding the computing power and energy efficiency issues.

Another possibility is appearance-based place recognition directly using video sequences [10]. Contrary to the VO/SLAM algorithms, the place recognition methods only determine if the observed place is similar to an already visited location. However, the place recognition methods scale better for large environments than typical SLAM algorithms [11]. In the appearance-based methods each image is processed and described by descriptors of salient features, or directly described by a vector of numerical values [12]. The FAB-MAP algorithm [13] organizes feature descriptors into the Bag of Visual Words [14], allowing efficient comparison of images using histogram-based methods. Recent approaches to direct image description involve usage of image sequences [15]. The use of sequences instead of individual images exploits the temporal coherence of vision data acquired by a moving camera [16], thus decreasing the number of false positives in place recognition for environments with self-similarities, and increasing the tolerance to local scene changes. This idea is used by the ABLE-M [17] algorithm, which substantially scales down the processed images, and compares global binary descriptors using the quick to compute Hamming distance. This makes ABLE-M an interesting choice for implementation on mobile devices. Although appearance-based place recognition techniques were already used on mobile devices [18], [19], our recent work [20] was probably the first one investigating an algorithm based on sequences of images on an actual smartphone.

### C. Contribution

Our experiments [20] revealed that the solution based on sequences is beneficial in corridor-like environments, where not enough salient features are available for reliable comparison based on a single query image. However, the processing time of the ABLE-M algorithm does not scale well for long database sequences. Therefore, we propose a much faster version of the ABLE-M algorithm, called FastABLE, that suits the needs of personal localization on mobile devices. In the improved version the processing time is independent from the size of the comparison window. The FastABLE code is publicly available as open source, together with the test sequences, to make our results fully verifiable.

M. Nowicki, J. Wietrzykowski and P. Skrzypczyński are with the Institute of Control and Information Engineering, Poznań University of Technology, ul. Piotrowo 3A PL-60-965 Poznań, Poland. Contact: [michal.nowicki@put.poznan.pl](mailto:michal.nowicki@put.poznan.pl)

<sup>1</sup><http://www.skyhookwireless.com/>

## II. INDOOR LOCALIZATION WITH OPENABLE

ABLE-M is the algorithm proposed by Arroyo et. al [17] that detects if the camera observes a place that was already visited, or determines if the currently observed scene is similar to a previously recorded image. The ABLE-M algorithm is available<sup>2</sup> in the open-source implementation OpenABLE [21]. This implementation allows to process a single video file, and determines if the given frame (image) depicts a place that was already present in this sequence. This is typical loop closure mode of operation, which is however impractical for personal indoor localization. We want to determine the user location in real-time by comparing a short video sequence acquired on the go, to a pre-recorded video of the environment. Therefore, our implementation of the system operates on two sequences: the first one (*train*) is recorded prior to the localization experiment, and can be annotated with respect to the known floor plan of the building, while the second one (*test*) is recorded on-line by the user and used as the query.

### A. Baseline algorithm

The ABLE-M algorithm processes sequences of images, thus the processing of individual frames has to be very fast to allow real-time operation. To this end ABLE-M resizes the processed images to the size of  $64 \times 64$  pixels. Next, a global variant of the LDB feature descriptor is computed with respect to the whole resized image. In contrary to LBP-based descriptors [22], the binary LDB descriptor [23] is computed by comparing individual intensities of pairs of points along the same lines. LDB was designed to suit the needs of mobile devices, thus it can be computed with minimal effort. It should be noted that prior to image description ABLE-M can perform the reduction of illumination changes in the sequence using the method of McManus *et al.* [24]. However, the illumination invariance is turned off by default in the OpenABLE implementation. As illumination changes in building interiors are limited, comparing it to outdoor scenes, we have decided to let this option be turned off, to make the processing faster.

The next step in the algorithm is the matching phase, in which the similarity matrix  $S$  between positions registered in the sequences *train* and *test* is computed. As the *train* sequence contains  $n$  global image descriptors, and the *test* sequence contains  $m$  descriptors, the matrix size is  $(m - c_l) \times (n - c_l)$ , and it is computed as:

$$S_{i,j} = \text{hamming}(\text{train}[i, i - c_l], \text{test}[j, j - c_l]), \quad (1)$$

where  $\text{hamming}(x, y)$  computes the Hamming distance between the descriptors of the sequences denoted as  $x$  and  $y$ . Finally, the resulting similarity matrix  $S$  is normalized. The notation  $\text{train}[i, i - c_l]$  represents the concatenation of descriptors from  $i$ -th to  $(i - c_l)$ -th frame of the *train* sequence, where  $c_l$  is the length of the sub-sequence used for matching (in [20] it was denoted *compareLength* and in [17] it was denoted  $d_{length}$ ). The baseline ABLE-M algorithm complexity is equal to  $\mathcal{O}(nmc_l)$ , as for each image in the recognition

sequence ( $m$ ) we slide the window  $n$  times and compare the sub-sequences in  $c_l$  operations.

### B. Processing time analysis of OpenABLE

An important aspect of mobile computing are the processing time and the related energy efficiency [3]. Thus, we analyze the OpenABLE implementation to figure out how to speed-up this system. The processing time taken by OpenABLE can be divided into two parts: one independent from the database sequence size (descriptors computation), and another one, strictly dependent on the database size (matching).

The experiments performed on a Samsung Galaxy Note 3 showed that the descriptor computation is an efficient operation. The image resizing takes approximately 0.5 ms per frame, while in our experiments the computation of global LDB descriptor took in average only 0.44 ms per image. The processing time taken by matching depends on the database size and the size of the comparison window ( $c_l$ ). In [20] we assumed that the video signal should be processed at 10 frames per second to achieve real-time localization. Therefore, the sizes of the database varying from 100 to 4000 images represent camera trajectories recorded during motion lasting from 10 to 400 seconds. The size of the comparison window was also chosen experimentally in [20], to avoid false positives due to the local self-similarity inside the building, and was varying from 20 to 80 frames. Results of experiments with several different settings for those parameters are presented in Fig. 1.

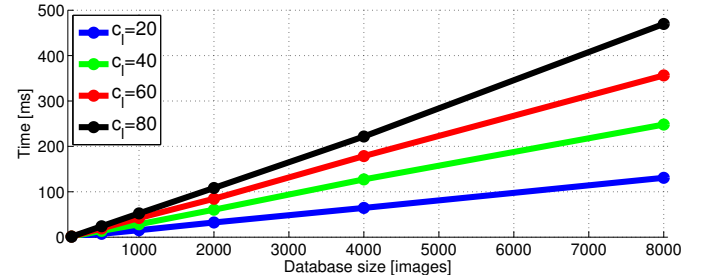


Fig. 1. Time taken to match a new sequence of images to the pre-recorded database depending on the database size and the length of comparison window for OpenABLE on Samsung Galaxy Note 3

From the plots in Fig. 1 it is evident that OpenABLE can be only applied in environments of limited size, as the matching times increase linearly not only with the growing size of the database, but also as a function of the comparison window size. The total processing time reaches 232 ms for  $c_l$  equal to 80, and the database size containing 4000 images. Further increase in the database size may prevent the system from operating in real-time. As we pointed out in [20], this might be an important factor when choosing between a typical place recognition algorithm (e.g. FAB-MAP) and OpenABLE.

## III. IMPROVED FASTABLE FOR MOBILE DEVICES

### A. Modifications to the algorithm

The OpenABLE implementation has several limitations when considering real-life applications on mobile devices. The processing results are presented in the form of a normalized

<sup>2</sup><https://github.com/roberto-arroyo/OpenABLE>

similarity matrix  $S$ . This poses problems for personal localization, as we usually require the system to inform about the recognized place as fast as it is detected. Therefore, we introduce a threshold  $d_t$  on the difference in the Hamming distance between the compared descriptors. The parameters  $d_t$  and  $c_l$  can be set manually to maximize the number of correct recognitions without false positives, but we propose to automatize this process. The parameter  $c_l$  should be large enough to ensure robustness to self-similarity of the environment, but cannot exceed the length of the shortest sequence used as query. Due to crossings inside the building, the pre-recorded video consists of  $p$  non-overlapping sequences. Knowing the value of  $c_l$  we match each of those sequences against the remaining  $p - 1$  pre-recorded sequences to find the minimal Hamming distance between the corresponding sub-sequences. As neither of the  $p$  parts overlap, no recognitions should be found, and  $d_t$  is computed for the  $a$ -th sequence:

$$d_t^a = \min_{i,j,a \neq b} \text{hamming}(\text{train}_a[i, i - c_l], \text{train}_b[j, j - c_l]), \quad (2)$$

where  $d_t^a$  is the distance threshold found for  $\text{train}_a$ , which is part  $a$  of the training sequence. This solution sets different thresholds for different parts of the database, as different areas in the environment can differ significantly in their appearance.

However, the most important modification in the FastABLE implementation aims at significantly increasing speed of the matching phase. Although ABLE-M can employ the FLANN library [25] for fast nearest neighbor search [17], this is an approximate technique, which improves processing time mainly for very long sequences, and is not implemented in OpenABLE. Thus, we propose a purely algorithmic modification, that enables the use of our system for real-time indoor localization on mobile devices. To introduce the idea, we analyze the steps taken during processing in ABLE-M (Fig. 2).

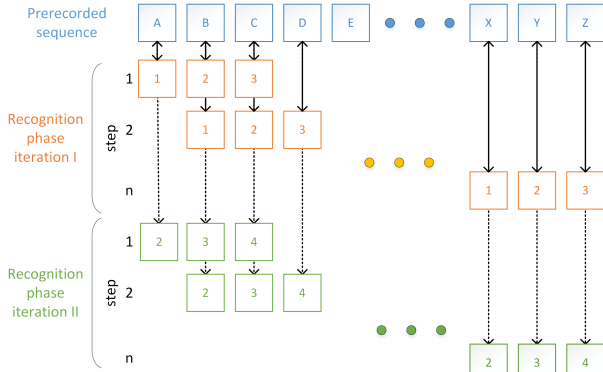


Fig. 2. After capturing the image, one iteration with  $n$  steps is performed to compare the sliding window of descriptors to the pre-computed descriptors from the database. In OpenABLE, each iteration is independent from the others. In FastABLE we utilize the previously computed values to speed-up the computations

In ABLE-M and the OpenABLE implementation, after a new image is captured, the window of descriptors computed on the last  $c_l$  images is moved along the pre-recorded sequence, and the corresponding Hamming distances are computed. The distance computations between the same pairs of global descriptors in the database sequence and the sliding window

are repeated. This can be observed in Fig. 2, as for example the distance between the descriptors  $B$  and 2 is computed in iteration 1 step 1, and in the iteration 2 step 2.

In contrary, in FastABLE we maximally re-use the information already computed. In the first iteration the corresponding distances are computed and saved in an auxiliary array  $prev$ . In the next iterations, we compute the distance for the first step the same way as in OpenABLE. However, in the following  $n - 1$  iterations it is possible to use the previously stored distance values. The distance  $d$  for the iteration  $m$  in the  $k$ -th step is computed as:

$$d_{m,k} = d_{m-1,k-1} + \text{hamming}(\text{train}[k + c_l], \text{test}[c_l]) - \text{hamming}(\text{train}[k - 1], \text{test}[1]), \quad (3)$$

where the indices are indexed from 1. The idea is to compute the current distance as the result of previous iteration reduced by the distance for the first element in the previous comparison, but increased by the comparison of the distance for the last element in the sliding window of the current iteration. As we manipulate integer Hamming distances, no residual errors accumulate due to the computations, and the results are numerically identical with those obtained from OpenABLE. In Fig. 2 the distance from the second step in second iteration,  $d_{2,2}$ , is the distance between substring  $\{B, C, D\}$  from pre-recorded sequence and substring  $\{2, 3, 4\}$  from the query sequence. Distance  $d_{2,2}$  according to FastABLE is computed by taking the distance  $d_{1,1}$  (distance between substring  $\{A, B, C\}$  and substring  $\{1, 2, 3\}$ ), subtracting the distance between  $A$  and 1 and increasing the result by the distance between  $D$  and 4. Each new distance updates the array  $prev$ , so that in the next iteration it is possible to re-use the previous results.

Comparing to algorithmic complexity of OpenABLE,  $\mathcal{O}(nmc_l)$ , the FastABLE complexity is equal to  $\mathcal{O}(nm)$  as we perform computations for each new images in the recognition sequence ( $m$ ), and we slide the window  $n$  times, but the comparison is performed in constant time ( $\mathcal{O}(1)$ ). Therefore, it is evident that the computational complexity of the modified algorithm is independent from the size of  $c_l$ . The computations are done at an increased memory cost to store the distances computed in previous iteration.

### B. Processing time analysis of FastABLE

The proposed algorithm was implemented and is available on GitHub in a version for PC<sup>3</sup>, and in a version for Android devices<sup>4</sup>. The experimental evaluation included comparison of the processing time for different  $c_l$  values and different sizes of the database. Results presented in Fig. 3 prove that the size of the sliding window  $c_l$  does not have an influence on the processing time anymore. Moreover, the obtained total processing times are much smaller than the times required to process the same data by OpenABLE. The maximal processing time took by FastABLE for a database of 4000 frames with  $c_l=80$  is below 7 ms, whereas OpenABLE needed 233 ms to accomplish the same task. Table I shows how many times faster is FastABLE with respect to OpenABLE for several database

<sup>3</sup><https://github.com/LRMPUT/FastABLE>

<sup>4</sup>[https://github.com/LRMPUT/FastABLE\\_Android](https://github.com/LRMPUT/FastABLE_Android)

sizes and  $c_l$  values. Gains due to the improved algorithm are especially evident for larger comparison window sizes.

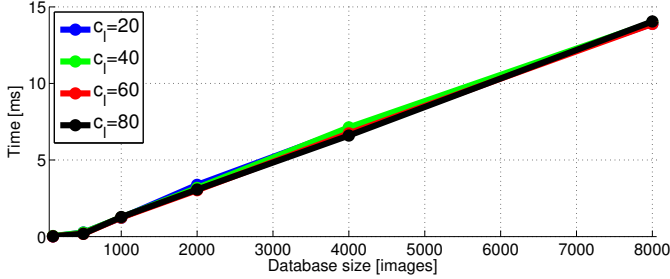


Fig. 3. Time taken to match a new sequence of images to the pre-recorded database depending on the database size and the length of comparison window for FastABLE on Samsung Galaxy Note 3

We evaluated FastABLE performance on the publicly available Nordland data set, used also to evaluate ABLE-M [17]. The data set contains video of a train ride, registered in four seasons. We compared sequences of the length 2500 frames (with  $c_l=300$  as in [17]) from the "fall" and "spring" videos using OpenABLE and FastABLE. The obtained similarity matrices  $S$  were identical for both systems, but OpenABLE required in average 0.068 ms for matching of two image subsequences 2, whereas FastABLE required only 0.00052 ms for this task.

TABLE I. PROCESSING TIME IMPROVEMENT OVER OPENABLE OBTAINED WITH FASTABLE

| compareLength ( $c_l$ ) | database size $n$ |       |      |      |      |      |
|-------------------------|-------------------|-------|------|------|------|------|
|                         | 100               | 500   | 1000 | 2000 | 4000 | 8000 |
| 20                      | 100.8             | 27.9  | 12.5 | 9.6  | 9.1  | 9.4  |
| 40                      | 40.0              | 53.6  | 22.7 | 19.0 | 17.8 | 17.7 |
| 60                      | 222.3             | 109.3 | 33.8 | 27.9 | 26.6 | 25.7 |
| 80                      | 54.7              | 135.2 | 41.0 | 35.3 | 33.6 | 33.4 |

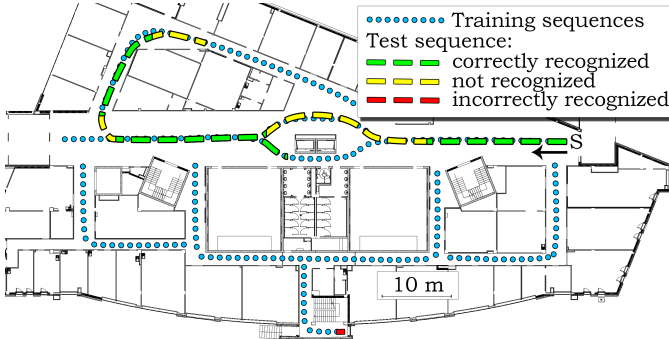


Fig. 4. FastABLE experimental evaluation in the PUT MC building. The 14 pre-recorded sequences (blue dots) cover corridors in both directions. The query path (rectangles) consists of mostly correctly recognized locations (green) with some parts not recognized (yellow), and few incorrect (red)

To verify the feasibility of FastABLE for indoor localization on mobile devices, we performed experiments on the first floor of the Poznan University of Technology Mechatronics Centre (PUT MC) building. The pre-recorded data consisted of 14 sequences recorded on non-overlapping paths in both directions of motion. The database amounted to 2127 images, while the query trajectory consisted of 320 images (Fig. 4). FastABLE reported 5625 correct place recognitions and only 80 incorrect recognitions, localizing the user on over a half of the trajectory, only with the camera images. Incorrect recognitions were

mainly found due to two, visually nearly identical paths around the elevators in the centre of the building. The results were obtained with automatically set parameters ( $c_l, d_t$ ), and could be further improved by manual tuning for the specific environment. For the reference implementation on PC/Linux and  $c_l=60$  the processing time for the whole query sequence in FastABLE was 0.76 s, with 0.0022 ms for matching two image sequences in average. OpenABLE needed 15.68 s to process the same data, being about 20 times slower. The same query sequence was processed by the OpenFABMAP implementation<sup>5</sup> of the FAB-MAP algorithm with minor modifications proposed in [20]. OpenFABMAP correctly recognized user locations at 6 places along the path from 30 training places (Fig. 5), but provided only sparse location information that might not be suitable for some applications. The sequence processing time was 15.60 s, whereas the matching of two image sequences took 39.78 ms in average.

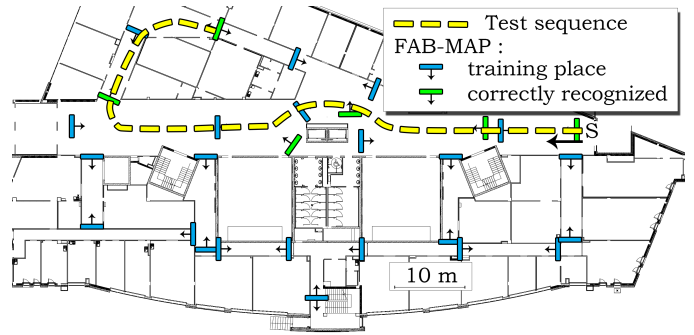


Fig. 5. OpenFABMAP experimental evaluation in the PUT MC building. From 30 training recognition places (blue), OpenFABMAP correctly recognized 6 query locations (green) along the test path

#### IV. CONCLUSIONS

The article presents modifications made to the indoor visual place recognition algorithm ABLE-M that allow to speed-up the computations over 20 times in typical indoor applications. An open source implementation of the improved algorithm is publicly available to allow other researchers to reproduce the presented results. Among the approaches to visual localization the appearance-based methods employing matching of video sequences suit well the specific requirements of personal localization with mobile devices. However, these methods require much more processing time for long pre-recorded sequences. Thus, localization in large buildings can become problematic. The proposed FastABLE solution allows to significantly increase the size of the environment that the localization system can handle in real-time, or to increase the energy efficiency for an environment of the same size. Our future research will focus on integrating FastABLE in the graph-based multi-sensor personal localization scheme for mobile devices we have introduced in [26].

#### ACKNOWLEDGMENT

This research was funded by the National Science Centre in Poland in years 2016-2019 under the grant 2015/17/N/ST6/01228.

<sup>5</sup><https://github.com/arenglover/openfabmap>

## REFERENCES

- [1] D. Scaramuzza and F. Fraundorfer. Visual Odometry [Tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92, Dec 2011.
- [2] T. Lemaire, C. Berger, I.-K. Jung, and S. Lacroix. Vision-Based SLAM: Stereo and Monocular Approaches. *International Journal of Computer Vision*, 74(3):343–364, 2007.
- [3] M. Fularz, M. Nowicki, and P. Skrzypczyński. *Adopting Feature-Based Visual Odometry for Resource-Constrained Mobile Devices*, pages 431–441. Springer International Publishing, Cham, 2014.
- [4] G. Klein and D. Murray. Parallel Tracking and Mapping on a camera phone. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 83–86, Oct 2009.
- [5] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, May 2014.
- [6] P. Tanskanen, T. Naegeli, M. Pollefeys, and O. Hilliges. Semi-direct EKF-based monocular visual-inertial odometry. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 6073–6078, Sept 2015.
- [7] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 298–304, Sept 2015.
- [8] M. A. Shelley. Monocular Visual Inertial Odometry on a Mobile Device. Master’s thesis, Institut für Informatik, TU München, Germany, 2014.
- [9] M. Quigley, D. Stavens, A. Coates, and S. Thrun. Sub-meter indoor localization in unmodified environments with inexpensive sensors. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2039–2046, Oct 2010.
- [10] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visual Place Recognition: A Survey. *IEEE Transactions on Robotics*, 32(1):1–19, Feb 2016.
- [11] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós. A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 2009.
- [12] Y. Liu and H. Zhang. Visual loop closure detection with a compact image descriptor. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1051–1056, Oct 2012.
- [13] M. Cummins and P. Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 2010.
- [14] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477 vol.2, Oct 2003.
- [15] P. Neubert, N. Sünderhauf, and P. Protzel. Appearance change prediction for long-term navigation across seasons. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 198–203, Sept 2013.
- [16] C. Piciarelli. Visual indoor localization in known environments. *IEEE Signal Processing Letters*, PP(99):1–1, 2016.
- [17] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera. Towards life-long visual localization using an efficient matching of binary sequences from images. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6328–6335, May 2015.
- [18] M. Nowicki. WiFi-guided visual loop closure for indoor localization using mobile devices. *Journal of Automation, Mobile Robotics & Intelligent Systems (JAMRIS)*, 8(3):10–18, 2014.
- [19] H. Wang, D. Zhao, H. Ma, H. Xu, and X. Hou. Crowdsourcing based mobile location recognition with richer fingerprints from smartphone sensors. In *Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on*, pages 156–163, Dec 2015.
- [20] M. Nowicki, J. Wietrzykowski, and P. Skrzypczyński. Experimental evaluation of visual place recognition algorithms for personal indoor localization. In *Indoor Positioning and Indoor Navigation (IPIN), 2016 International Conference on*, Oct. 2016.
- [21] R. Arroyo, L. M. Bergasa, and E. Romera. OpenABLE: An Open-source Toolbox for Application in Life-Long Visual Localization of Autonomous Vehicles. In *2016 Intelligent Transportation Systems Conference (ITSC)*, Nov. 2016.
- [22] J. Ren, X. Jiang, and J. Yuan. LBP Encoding Schemes Jointly Utilizing the Information of Current Bit and Other LBP Bits. *IEEE Signal Processing Letters*, 22(12):2373–2377, Dec 2015.
- [23] X. Yang and K.-T. Cheng. Ldb: An ultra-fast feature for scalable augmented reality on mobile devices. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 49–57, Nov 2012.
- [24] C. McManus, W. Churchill, W. Maddern, A. D. Stewart, and P. Newman. Shady dealings: Robust, long-term visual localisation using illumination invariance. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 901–906, May 2014.
- [25] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014.
- [26] M. Nowicki and P. Skrzypczyński. *Indoor Navigation with a Smartphone Fusing Inertial and WiFi Data via Factor Graph Optimization*, pages 280–298. Springer International Publishing, 2015.